MORE FUNCTIONS, DECISION STRUCTURES

CSSE 120—Rose Hulman Institute of Technology

Checkout Session08Robotics project

- Into Eclipse…

Questions on concepts on Exam 1

- Last session's slides contained a list of concepts that will be on exam 1
- What questions on these concepts have come up so far as you study for the exam?

- Any other questions?

- Notes:
  - Today's material will not be on the exam.
  - The homework assigned today will be due Session 10 (Tuesday after break)

Function Review

- Functions can take multiple parameters
  - distance(p1, p2)
- Functions can return values

```
def square(x):
    return x * x
```

- More about parameters:
  - What happens when we modify them?
  - What is an optional parameter?
- More about return values:
  - Can return multiple values

Passing parameters in Python

- What type of information do formal parameters receive?
- If we assign new values to formal parameters, does this affect the actual parameters?
- Consider this version of square:

```
def squareNext(x):
    x = x + 1
    return  x * x
```

Optional parameters

- A python function may have some parameters that are optional.

```
>>> int("37")
37
>>> int("37", 10)
37
>>> int("37", 8) # specify base 8
31
```

We can declare a parameter to be optional by supplying a default value.

```
>>> def printDate(month, day, year=2007):
        print month, str(day)+",", year


>>> printDate("January", 4, 2006)
January 4, 2006
>>> printDate("January", 4)
January 4, 2007
```

Q4

Multiple Value Return

- A function can return multiple values
  - **def powers(n):**
    return n2, n**3, n**4**
- What's the type of the value returned by this:
  powers(4)

Decision, Decisions

- Sometimes we have to alter the sequential flow of a program
  - What examples have we seen of this?
- Statements that alter the flow are called

  control structures
- *Decision structures* are control structures that allow programs to "choose" between different sequences of instructions

Sensors (from Pycreate)

- buttons = robot.getSensor(*"BUTTONS"*)

BUTTONS	The state of the Play and Advance buttons on the Create (0 = button not pressed, 1 = button pressed). Interpreted as an array of 2 values: [Advance, Play]

- *buttons = robot.getSensor("BUMPS_AND_WHEEL_DROPS")*

BUMPS_AND_WHEEL_DROPS	The values of the bumper and wheel drop sensors (0 = no bump, 1 = bump or 0 = wheel raised, 1 = wheel dropped). Interpreted as an array of 5 values: [Wheeldrop Caster, Wheeldrop Left, Wheeldrop Right, Bump Left, Bump Right]

Simple Decisions

- The **if** statement
  - if <condition>:
    <body>
  - Semantics:
    "if the condition is true, run the body, otherwise skip it"
- Simple conditions
  - <expr> <relop> <expr>
  - Some relational operators:

Math	<	≤	=	≥	>	≠
Python	<	<=	==	>=	>	!=

Class Exercise

- Modify sense.py so that it prints “Advance” whenever the advance button is pressed and “Play” whenever the play button is pressed.

- buttons = robot.getSensor(“*BUTTONS*”)

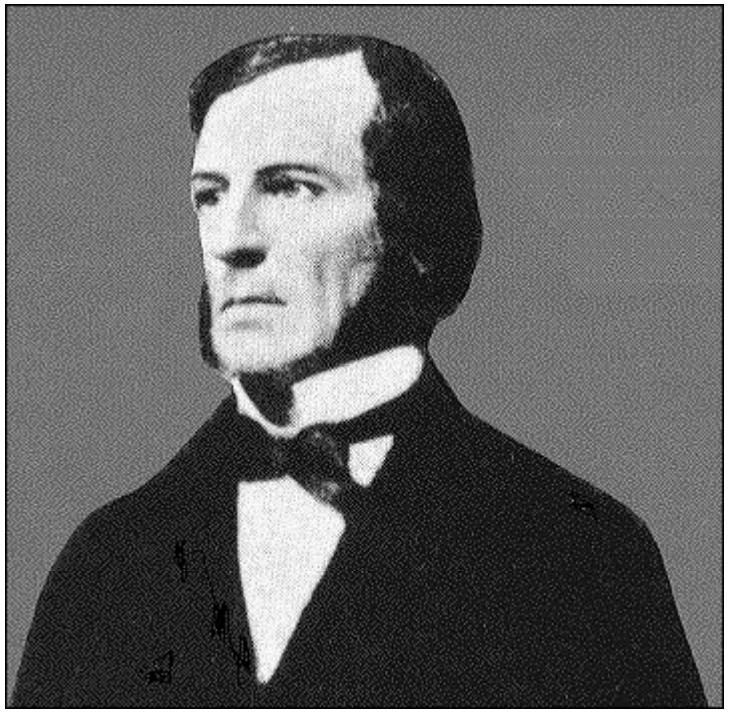
BUTTONS	The state of the Play and Advance buttons on the Create (0 = button not pressed, 1 = button pressed). Interpreted as an array of 2 values: [Advance, Play]

More on Comparisons

- Conditions are *boolean expressions*
  - They evaluate to True or False
- Try in IDLE:

```
>>> 3 < 4
>>> 42 > 7**2
>>> "ni" == "Ni"
>>> "A" < "B"
>>> "a" < "B“
```

- >>> 3 < 4 and 5 < 6
- >>> 3 < 4 and 5 < 3



George Boole

Having It Both Ways: if-else

- Syntax:

  if <condition>:

  <statementsForTrue>

  else:

  <statementsForFalse>

- Semantics:

  "If the condition is true, execute the statements for true, otherwise execute the statements for false"

A Mess of Nests

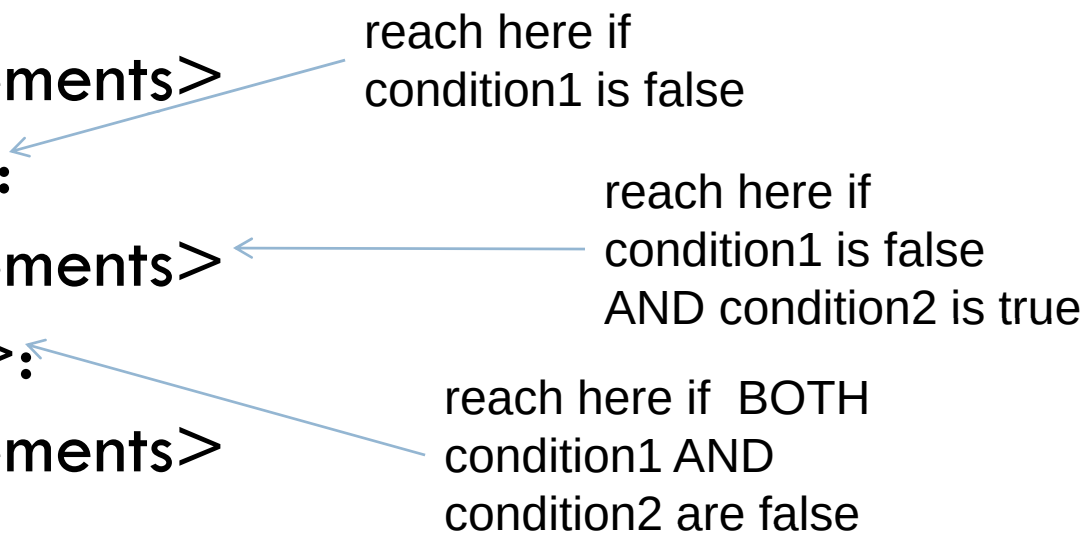
- Consider using the bump sensors to detect collisions with other obstacles

- Could we modify our program so that it can detect the difference between:
  - a head-on collision (both left and right bumpers pressed)
  - A glancing blow on the left (left bumper only)
  - A glancing blow on the right (right bumper only)
  - No collision

Multi-way Decisions

- Syntax:

```
if <condition1>:
        <case 1 statements>
elif <condition2>:
        <case 2 statements>
elif <condition 3>:
        <case 3 statements>
…
else:
        <default statements>
```



Cleaning the Bird Cage

- Advantages of **if-elif-else** vs. nesting
  - Number of cases is clear
  - Each parallel case is at same level in code
  - Less error-prone

- Code together the logic to distinguish between:
  - a head-on collision (both left and right bumpers pressed)
  - A glancing blow on the left (left bumper only)
  - A glancing blow on the right (right bumper only)
  - No collision

Q8

Individual Exercise on Using if-else

- Finish the quiz first.

- Then open **countPassFail.py**.
- Define (in that file) a function **countPassFail(scores)** that
  - takes a list of exam scores
  - *returns* two values:
    - the count of passing scores in the list (those at least 60), and
    - the count of failing scores in the list
- Examples:
  - **print countPassFail([57, 100, 34, 87, 74])** prints **(3,2)**
  - **print countPassFail([59])** prints **(0,1)**
  - **print countPassFail([])** prints **(0,0)**
- Commit your project to your repository.